What is claimed is:

1. (Currently Amended) A computer-implemented method for querying a structured document, comprising:

identifying auxiliary structures including pre-computed information applicable to accelerate user query processing by detecting containment mappings between query expressions and expressions in the auxiliary structures,

wherein the auxiliary structures include a number of indexes, a number of partial XML indexes, and a number of materialized views;

computing compensation ~~for~~ to perform index selection ~~or~~ and materialized view matching to determine what portion of said query expressions ~~can be~~ are evaluated by said index and said materialized view; ~~and~~

finding the user query result by executing a rewritten query that exploits the pre-computed information ~~for~~ to each detected containment mapping; and

reporting said user query result to said user.

2. (Original) The method of claim 1 further comprising implementing the method in a relational database management system.

3. (Original) The method of claim 1 wherein the structured document includes a set of nodes described by an expression tree.

4. (Original) The method of claim 1 wherein the structured document is an XML document.

5. (Cancelled).

6. (Original) The method of claim 1 wherein the pre-computed information includes pre-computed XPath results (PXRs).

7. (Original) The method of claim 1 wherein the user query processing further comprises navigating path expressions with a query language.

8. (Original) The method of claim 7 wherein the query language employs XPath.

9. (Original) The method of claim 7 wherein the query language includes at least one of: XQuery, SQL/XML, and XSLT.

10. (Currently Amended) The method of claim 1 wherein the detecting further comprises:

selectively executing a set of predetermined sequential rules ~~for~~ to perform traversing of a tree of nodes; matching node data with the pre-computed information; and

selecting auxiliary structures that subsume portions of the user query.

11. (Original) The method of claim 10 wherein the node data includes axis data, test data, predicate data, and next step node data.

12. (Original) The method of claim 10 further comprising normalizing expression trees by moving predicate conditions into filter expressions before the identifying.

13. (Currently Amended) The method of claim 1 wherein executing the rewritten query further comprises:

constructing a pushdown expression ~~for~~ to perform evaluation with information in the auxiliary structure; and

constructing a compensation expression ~~for~~ to perform evaluation as a residual query, wherein said residual query comprises at least a portion of said query not associated with said index or said materialized view.

14. (Original) The method of claim 13 wherein the compensation expression is an XPath predicate.

15. (Original) The method of claim 13 further comprising building a taxonomy of auxiliary structures.

16. (Currently Amended) The method of claim 15 further comprising classifying compensation expressions ~~for~~ to the taxonomy according to a predetermined set of values.

17. (Original) The method of claim 1 wherein the identifying handles at least one of: nested path expressions, nested predicates, value-based comparison predicates, conjunction, disjunction, all XPath axes, branches, and wild cards.

18. (Original) The method of claim 17 wherein the XPath axes include child, descendant, self, attribute, parent, and descendant-or-self.

19. (Original) The method of claim 1 further comprising creating a mapping directed acyclic graph (DAG) that separately encodes a set of all containment mappings for each node.

20. (Previously Presented) The method of claim 19 wherein creating the mapping DAG is polynomial in terms of a size of expression trees.

21. (Original) The method of claim 19 further comprising pruning the mapping DAG to remove invalid node pairs.

22. (Currently Amended) A computer-based system for querying a structured document, comprising:

an identifier of auxiliary structures including pre-computed information applicable to accelerate user query processing by detecting containment mappings between query expressions and expressions in the auxiliary structures, wherein the auxiliary structures include a number of indexes, a number of partial XML indexes, and a number of materialized views;

a computer that computes computing compensation ~~for~~ to perform index selection ~~or~~ and materialized view matching to determine what portion of said query expressions ~~can be~~ are evaluated by said index and said materialized view; and

a query evaluator that finds the user query result by executing a rewritten query that exploits the pre-computed information ~~for~~ to each detected containment mapping, wherein said query evaluator reports said user query result to said user.

23. (Original) The system of claim 22 that is implemented in a relational database management system.

24. (Original) The system of claim 22 wherein the structured document includes a set of nodes described by an expression tree.

25. (Original) The system of claim 22 wherein the structured document is an XML document.

26. (Cancelled).

27. (Original) The system of claim 22 wherein the pre-computed information includes pre-computed XPath results (PXRs).

28. (Original) The system of claim 22 wherein the user query processing employs a query language that navigates path expressions.

29. (Original) The system of claim 28 wherein the query language employs XPath.

30. (Original) The system of claim 28 wherein the query language includes at least one of: XQuery, SQL/XML, and XSLT.

31. (Currently Amended) The system of claim 22 wherein the identifier:

selectively executes a set of predetermined sequential rules ~~for~~ to perform traversing of a tree of nodes; matches node data with the pre-computed information; and

selects auxiliary structures that subsume portions of the user query.

32. (Original) The system of claim 31 wherein the node data includes axis data, test data, predicate data, and next step node data.

33. (Original) The system of claim 31 wherein the identifier normalizes expression trees by moving predicate conditions into filter expressions before the identifier begins detecting.

34. (Currently Amended) The system of claim 22 wherein executing the rewritten query further comprises:

constructing a pushdown expression ~~for~~ to perform evaluation with information in the auxiliary structure; and

constructing a compensation expression ~~for~~ to perform evaluation as a residual query, wherein said residual query comprises at least a portion of said query not associated with said index or said materialized view.

35. (Original) The system of claim 34 wherein the compensation expression is an XPath predicate.

36. (Original) The system of claim 34 wherein the identifier builds a taxonomy of auxiliary structures.

37. (Currently Amended) The system of claim 36 wherein the identifier classifies compensation expressions ~~for~~ to the taxonomy according to a predetermined set of values.

38. (Original) The system of claim 22 wherein the identifier handles at least one of: nested path expressions, nested predicates, value-based comparison predicates, conjunction, disjunction, all XPath axes, branches, and wild cards.

39. (Original) The system of claim 38 wherein the XPath axes include child, descendant, self, attribute, parent, and descendant-or-self.

40. (Original) The system of claim 22 wherein the identifier creates a mapping directed acyclic graph (DAG) that separately encodes a set of all containment mappings for each node.

41. (Previously Presented) The system of claim 40 wherein creating the mapping DAG is polynomial in terms of a size of expression trees.

42. (Original) The system of claim 40 wherein the identifier prunes the mapping DAG to remove invalid node pairs.

43. (Currently Amended) A computer program product tangibly embodying a program of computer-executable instructions to perform a method for querying a structured document, the method comprising:

identifying auxiliary structures including pre-computed information applicable to accelerate user query processing by detecting containment mappings between query expressions and expressions in the auxiliary structures,

wherein the auxiliary structures include a number of indexes, a number of partial XML indexes, and a number of materialized views;

computing compensation ~~for~~ to perform index selection ~~or~~ and materialized view matching to determine what portion of said query expressions ~~can be~~ are evaluated by said index and said materialized view; ~~and~~

finding the user query result by executing a rewritten query that exploits the pre-computed information ~~for~~ to each detected containment mapping; and

<u>reporting said user query result to said user</u>.

44. (Original) The computer program product of claim 43 further comprising implementing the method in a relational database management system.

45. (Original) The computer program product of claim 43 wherein the structured document includes a set of nodes described by an expression tree.

46. (Original) The computer program product of claim 43 wherein the structured document is an XML document.

47. (Cancelled).

48. (Original) The computer program product of claim 43 wherein the pre-computed information includes pre-computed XPath results (PXRs).

49. (Original) The computer program product of claim 43 wherein the user query processing further comprises navigating path expressions with a query language.

50. (Original) The computer program product of claim 49 wherein the query, language employs XPath.

51. (Original) The computer program product of claim 49 wherein the query language includes at least one of: XQuery, SQL/XML, and XSLT.

52. (Currently Amended) The computer program product of claim 43 wherein the detecting further comprises:

selectively executing a set of predetermined sequential rules ~~for~~ to perform traversing of a tree of nodes;

matching node data with the pre-computed information; and

selecting auxiliary structures that subsume portions of the user query.

53. (Original) The computer program product of claim 52 wherein the node data includes axis data, test data, predicate data, and next step node data.

54. (Original) The computer program product of claim 52 further comprising normalizing expression trees by moving predicate conditions into filter expressions before the identifying.

55. (Currently Amended) The computer program product of claim 43 wherein executing the rewritten query further comprises:

constructing a pushdown expression ~~for~~ to perform evaluation with information in the auxiliary structure; and constructing a compensation expression ~~for~~ to perform evaluation as a residual query, wherein said residual query comprises at least a portion of said query not associated with said index or said materialized view.

56. (Original) The computer program product of claim 55 wherein the compensation expression is an XPath predicate.

57. (Original) The computer program product of claim 55 further comprising building a taxonomy of auxiliary structures.

58. (Currently Amended) The computer program product of claim 57 further comprising classifying compensation expressions ~~for~~ to the taxonomy according to a predetermined set of values.

59. (Original) The computer program product of claim 43 wherein the identifying handles at least one of:

nested path expressions, nested predicates, value-based comparison predicates, conjunction, disjunction, all XPath axes, branches, and wild cards.

60. (Original) The computer program product of claim 59 wherein the XPath axes include child, descendant, self, attribute, parent, and descendant-or-self.

61. (Original) The computer program product of claim 43 further comprising creating a mapping directed acyclic graph (DAG) that separately encodes a set of all Containment mappings for each node.

62. (Previously Presented) The computer program product of claim 61 wherein creating the mapping DAG is polynomial in terms of a size of expression trees.

63. (Original) The computer program product of claim 61 further comprising pruning the mapping DAG to remove invalid node pairs.

64. (Currently Amended) A system comprising:

means for identifying auxiliary structures including pre-computed information applicable to accelerate user query processing by detecting containment mappings between query expressions and expressions in the auxiliary structures,

wherein the auxiliary structures include a number of indexes, a number of partial XML indexes, and a number of materialized views;

means for computing compensation ~~for~~ to perform index selection ~~or~~ and materialized view matching to determine what portion of said query expressions ~~can be~~ are evaluated by said index and said materialized view; ~~and~~

means for finding the user query result by executing a rewritten query that exploits the precomputed information ~~for~~ to each detected containment mapping; and

means for reporting said user query result to said user.

65. (New) The method of claim 1, wherein said detecting of said containment mappings between said query expressions and said expressions in said auxiliary structures comprises matching node data of said query expressions with XPath expressions of said index and said materialized view, wherein said node data comprises axis data, test data, predicate data, and next XPath step node data.

66. (New) The system of claim 22, wherein said detecting of said containment mappings between said query expressions and said expressions in said auxiliary structures comprises matching node data of said query expressions with XPath expressions of said index and said materialized view, wherein said node data comprises axis data, test data, predicate data, and next XPath step node data.

67. (New) The computer program product of claim 43, wherein said detecting of said containment mappings between said query expressions and said expressions in said auxiliary structures comprises matching node data of said query expressions with XPath expressions of said index and said materialized view, wherein said node data comprises axis data, test data, predicate data, and next XPath step node data.